Iterative Linear Quadratic Regulator

Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

LQR is cute…

But what if my

robot is not linear?


EVERY SINGLE THING ON EARTH IS
EITHER BANANAS
OR NOT BANANAS
made with mematic

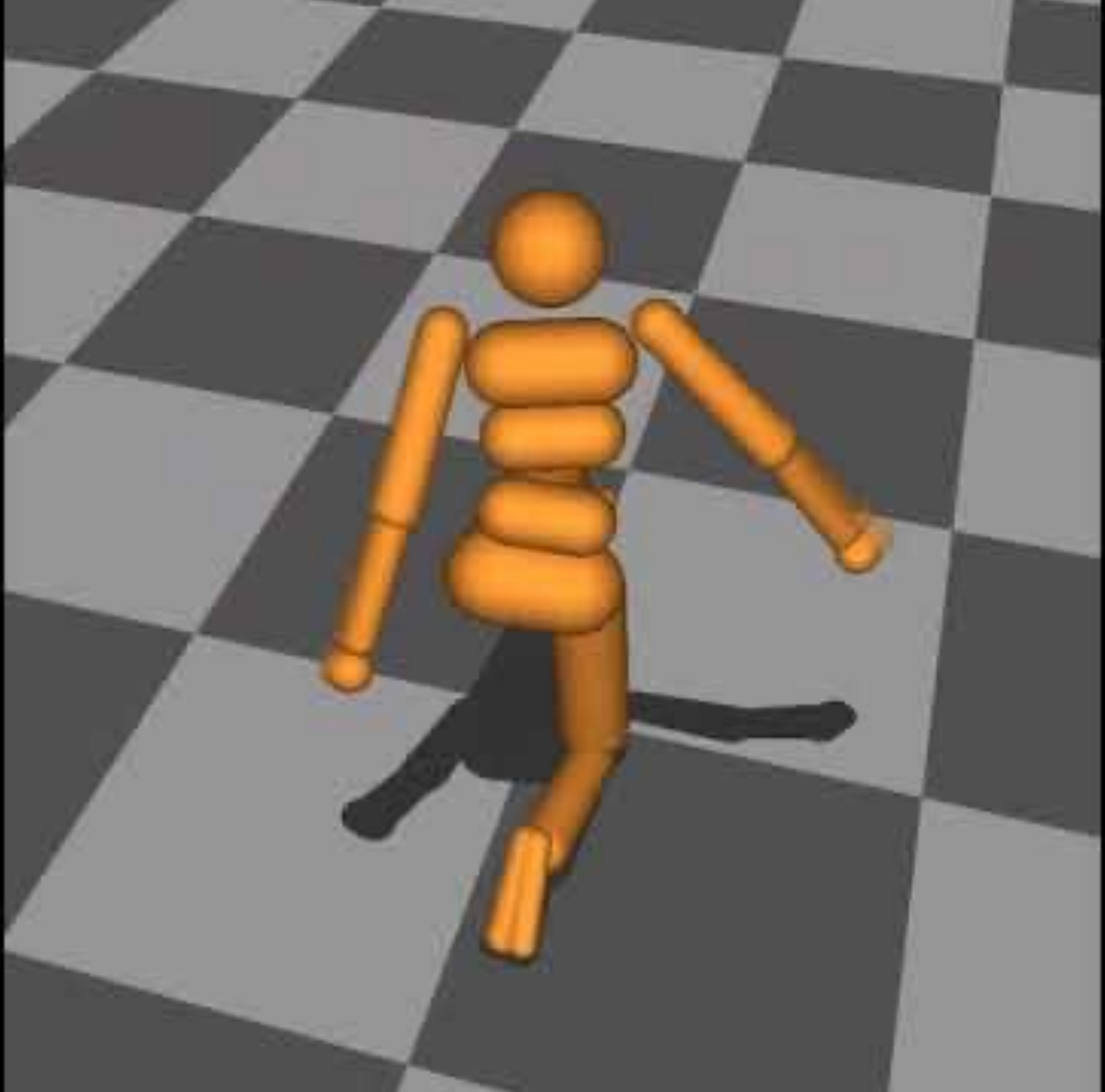
Two concerns?

Concern 1: Is LQR optimal for non-linear / non-quadratic costs?
If not, does it totally fall apart?

Simulation Lemma: If the model has $O(\epsilon)$*,*
the optimal policy for the model will have $O(\epsilon T^2)$ *suboptimality*

Concern 2: If LQR is suboptimal, what's the point of using it?

LQR is
fundamentally a way
to
locally approximate
and
update value functions



(Super cool work by Pieter Abeel et al. https://people.eecs.berkeley.edu/~pabbeel/autonomous_helicopter.html)

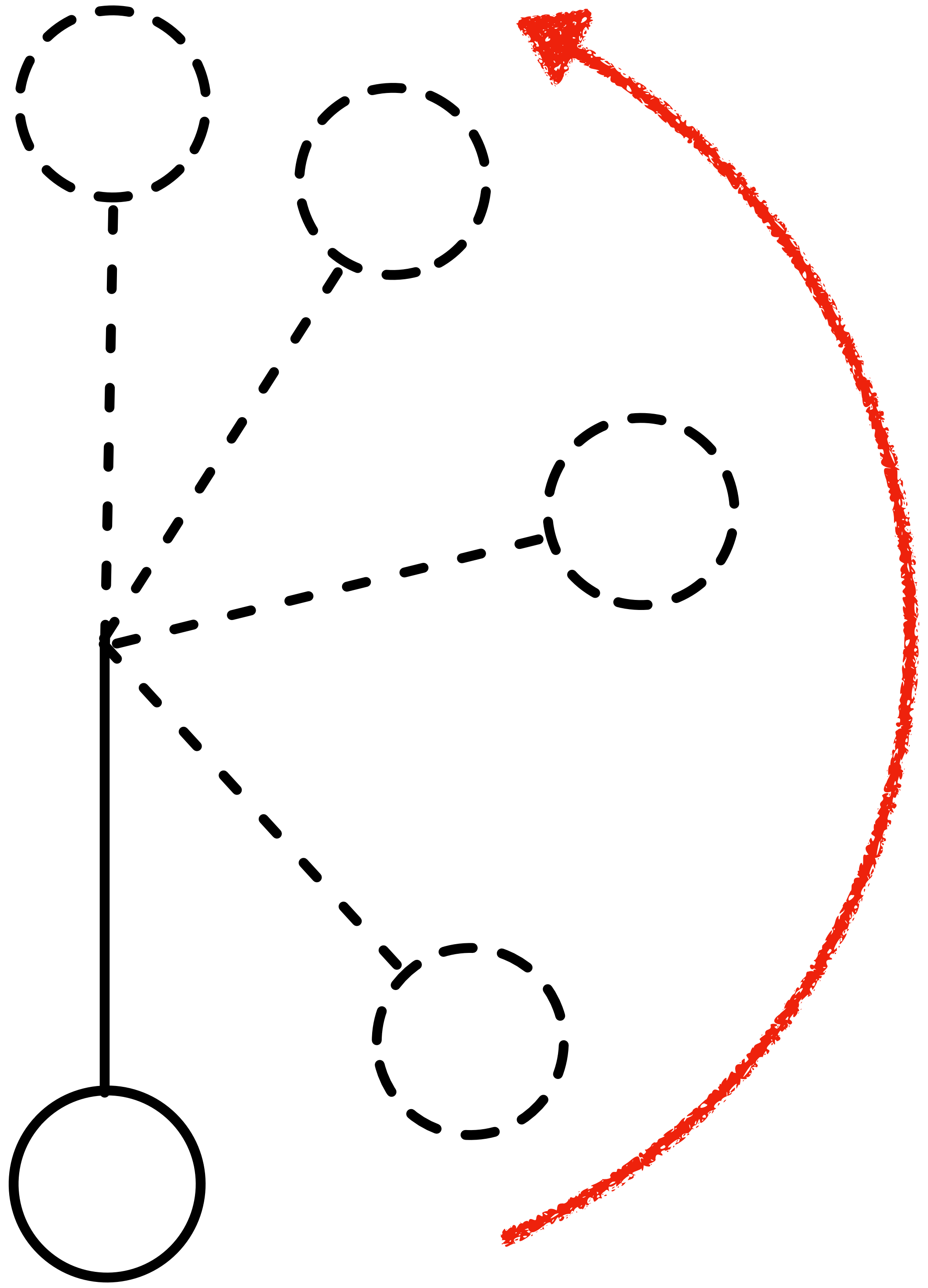
Activity!

Think-Pair-Share!

Think (30 sec): How can we use LQR to swing up a pendulum and stabilize it there?

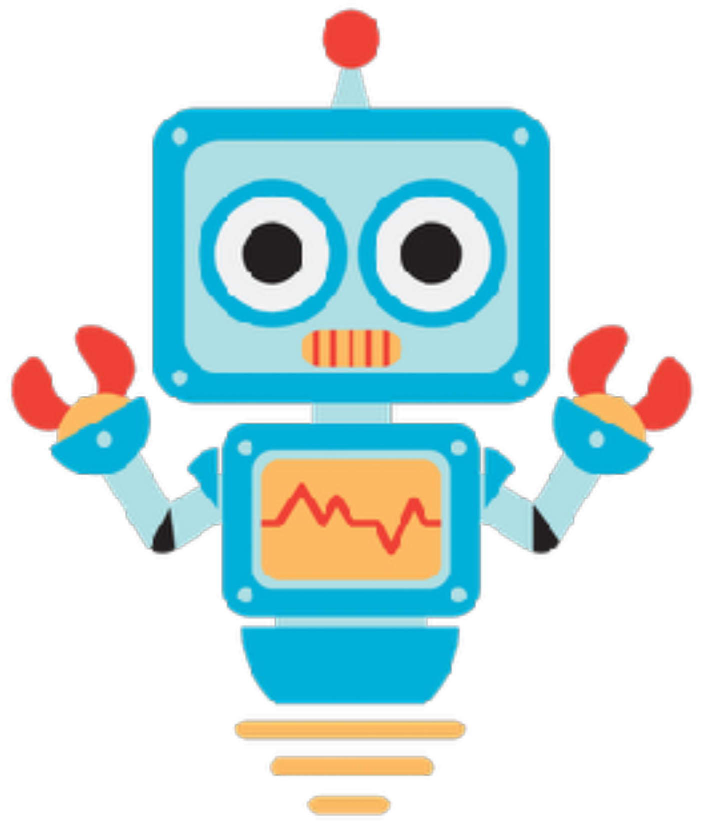
Pair: Find a partner

Share (45 sec): Partners exchange ideas



Iterative LQR (ILQR)

Goal: Solve a *general* continuous time MDP



Nonlinear!

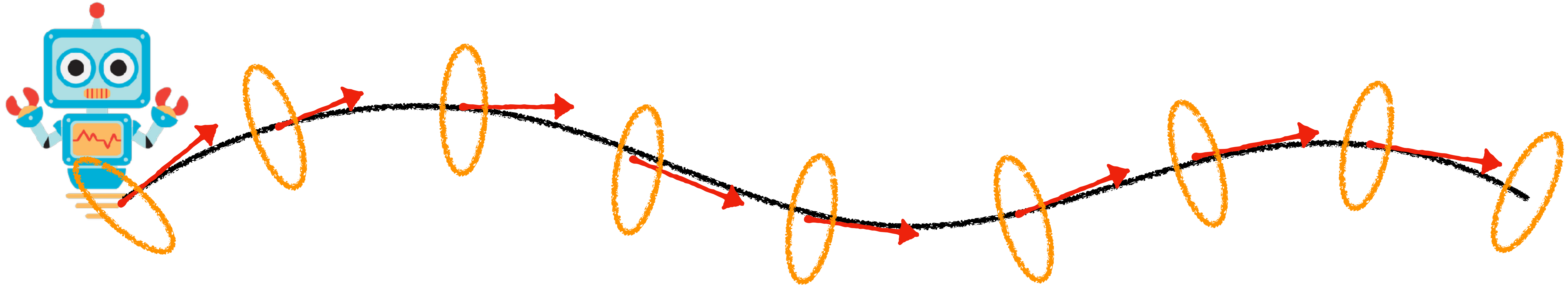
$$\min_{x_{0:T-1}, u_{0:T-1}} \sum_{t=0}^{T-1} c(x_t, u_t)$$

$$x_{t+1} = f(x_t, u_t)$$

Nonlinear!

Iterative LQR (ILQR) - Spill the beans!

Three simple steps!



Step 1: Forward pass - roll out current guess $u(t)$

Step 2: Linearize dynamics, quadricize cost around roll out

Step 3: Backwards pass - compute LQR gains K_t at each time

How I learned ILQR ..

Suffer through a barrage of matrix derivations!

(And god forbid you flip a sign...)

Dynamic Programming (Value-Iteration) Backup

Assume we have now a control policy of the form of a "feedforward" update term k_t and feedback term K_T that is a linear controller response to "errors" in z_T:

$$v_T = K_T z_T + k_T, \tag{2.7.1}$$

Inductively, we assume the next-state value function (i.e. of the future timestep) can be written in the form,

$$J_{T+1} = \frac{1}{2} z_{T+1} V_{T+1} z_{T+1} + G_{T+1} z_{T+1} + W_{T+1}. \tag{2.7.2}$$

Since

$$\begin{aligned} z_{T+1} &= A z_t + B v_t && (2.7.3)\\ &= A z_T + B(K_T z_T + k_T) && (2.7.4)\\ &= (A + BK_T) z_T + B k_T, && (2.7.5) \end{aligned}$$

we can write, J_{T+1} as:

$$\begin{aligned} J_{T+1} &= \frac{1}{2}((A+BK_T)z_T + Bk_T)^T V_{T+1}((A+BK_T)z_T + Bk_T) + G_{T+1}((A+BK_T)z_T + Bk_T) + W_{T+1} && (2.7.6)\\ &= \frac{1}{2} z_T^T (A+BK_T)^T V_{T+1}(A+BK_T) z_T + \frac{1}{2} k_T^T B^T V_{T+1} B k_T + k_T^T B^T V_{T+1}(A+BK_T) z_T && (2.7.7)\\ &\quad + C_{T-1}(A+BK_T) z_T + C_{T-1} B k_T + W_{T-1} && (2.7.8)\\ &= \frac{1}{2} z_T^T (A+BK_T)^T V_{T+1}(A+BK_T) z_T + \left(k_T^T B^T V_{T+1}(A+BK_T) + G_{T+1}(A+BK_T)\right) z_T && (2.7.9)\\ &\quad + G_{T-1} B k_T + \frac{1}{2} k_T^T B^T V_{T-1} B k_T + W_{T-1} && (2.7.10) \end{aligned}$$

Additionally, we can write the cost $c_T(z_T, v_T)$ as:

$$\begin{aligned} c_T &= \frac{1}{2} z_T^T Q z_T + z_T^T P v_T + \frac{1}{2} v_T^T R v_T + g_x^T z_T + g_u^T v_T + c + J_{T+1} && (2.7.11)\\ &= \frac{1}{2} z_T^T Q z_T + z_T^T P(K_T z_T + k_T) + \frac{1}{2}(K_T z_T + k_T)^T R (K_T z_T + k_T) + g_x^T z_T + g_u^T (K_T z_T + k_T) + c && (2.7.12)\\ &= \frac{1}{2} z_T^T Q z_T + z_T^T P K_T z_T + k_T^T P^T z_T + \frac{1}{2} z_T^T K_T^T R K_T z_T + \frac{1}{2} k_T^T R k_T + k_T^T R K_T z_T + g_x^T z_T && (2.7.13)\\ &\quad + g_u^T K_T z_T + g_u^T k_T + c \\ &= \frac{1}{2} z_T^T \left(Q + 2PK_T + K_T^T R K_T\right) z_T + \left(k_T^T P^T + k_T^T R K_T + g_x^T + g_u^T K_T\right) z_T + \frac{1}{2} k_T^T R k_T + g_u^T k_T + c && (2.7.14) \end{aligned}$$

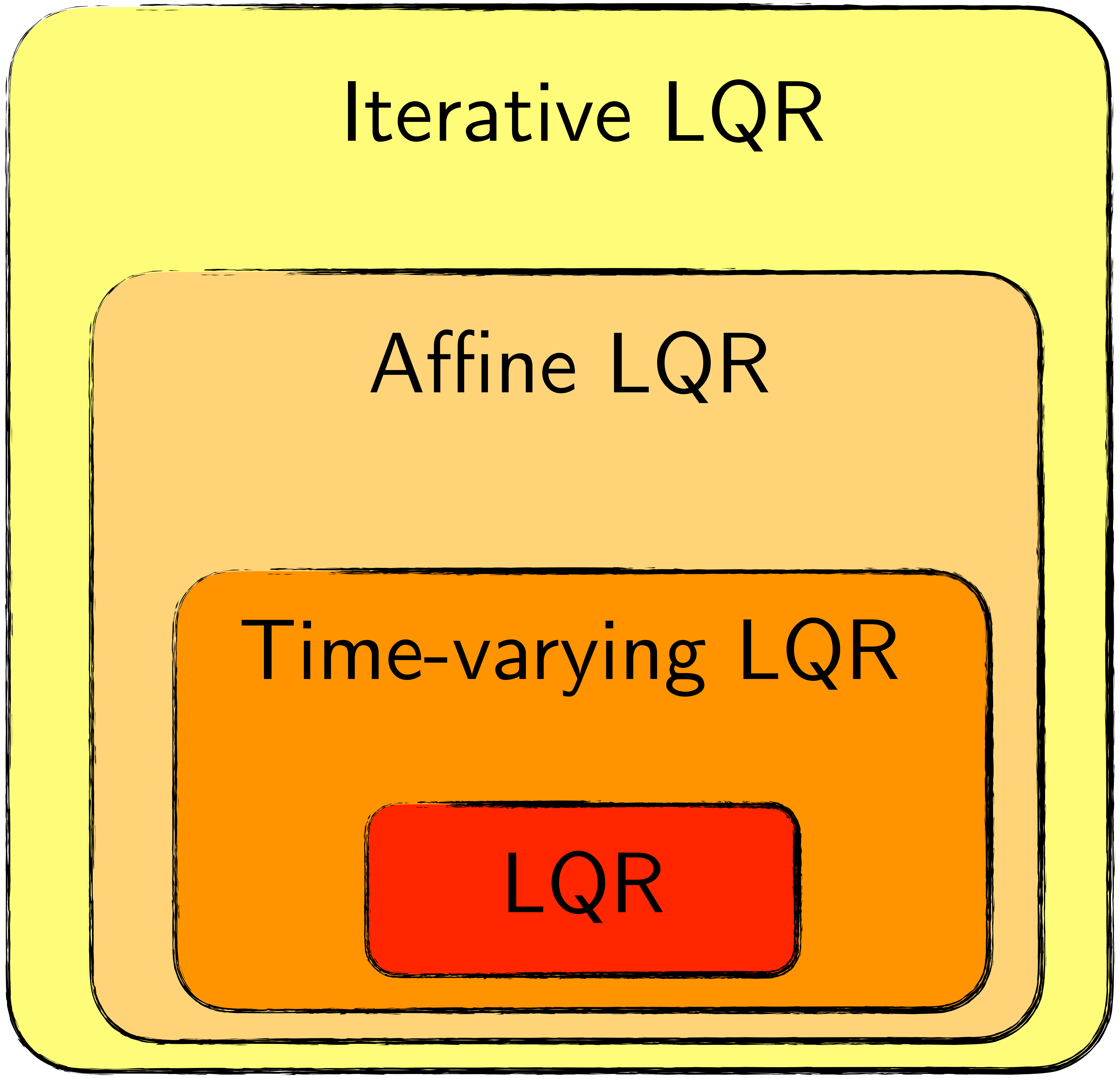
Then, we can write $J_T = c_T(z_T, v_T) + J_{T+1} = \frac{1}{2} z_T^T V_T z_T + G_T z_T + W_T$ by combining like terms from above, where

$$\begin{aligned} V_T &= Q + 2PK_T + K_T^T R K_T + (A+BK_T)^T V_{T+1}(A+BK_T) && (2.7.15)\\ G_T &= k_T^T P^T + k_T^T R K_T + g_x^T + g_u^T K_T + k_T^T B^T V_{T+1}(A+BK_T) + G_{T+1}(A+BK_T) && (2.7.16)\\ W_T &= \frac{1}{2} k_T^T R k_T + g_u^T k_T + c + G_{T+1} B k_T + \frac{1}{2} k_T^T B^T V_{T-1} B k_T + W_{T-1} && (2.7.17) \end{aligned}$$

We find the control policy by minimizing J_T with respect to v_T.

$$\begin{aligned} v_T &= \min_{v_T} c_T + J_{T+1} && (2.7.18)\\ &= z_T^T P v_T + \frac{1}{2} v_T^T R v_T + g_u^T v_T + \frac{1}{2}(A z_T + B v_T)^T V_{T+1}(A z_T + B v_T) + G_{T+1}(A z_T + B v_T) && (2.7.19)\\ &= \left(z_T^T P + z_T^T A^T V_{T+1} B\right) v_T + (G_{T+1} B + g_u^T) v_T + \frac{1}{2} v_T^T \left(R + B^T V_{T+1} B\right) v_T && (2.7.20)\\ & && (2.7.21) \end{aligned}$$

Strategy: Build up on LQR



$$x_{t+1} = \left.\frac{\partial f}{\partial x}\right|_{x_t} \delta x_t + \left.\frac{\partial f}{\partial u}\right|_{u_t} \delta u_t + f(x_t^*, u_t^*)$$

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}$$

$$x_{t+1} = A_t x_t + B_t u_t$$

$$x_{t+1} = A x_t + B u_t$$

Strategy: Build up on LQR

Iterative LQR

Affine LQR

Time-varying LQR

LQR

$$x_{t+1} = \frac{\partial f}{\partial x}\bigg|_{x_t} \delta x_t + \frac{\partial f}{\partial u}\bigg|_{u_t} \delta u_t + f(x_t^*, u_t^*)$$

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}$$

$$x_{t+1} = A_t x_t + B_t u_t$$

✔ $x_{t+1} = Ax_t + Bu_t$

Strategy: Build up on LQR

Iterative LQR

Affine LQR

Time-varying LQR

LQR

$$x_{t+1} = \frac{\partial f}{\partial x}\Big|_{x_t} \delta x_t + \frac{\partial f}{\partial u}\Big|_{u_t} \delta u_t + f(x_t^*, u_t^*)$$

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}$$

$$x_{t+1} = A_t x_t + B_t u_t$$

✓ $x_{t+1} = Ax_t + Bu_t$

Strategy: Build up on LQR

Iterative LQR

Affine LQR

Time-varying LQR

LQR

$$x_{t+1} = \frac{\partial f}{\partial x}\bigg|_{x_t} \delta x_t + \frac{\partial f}{\partial u}\bigg|_{u_t} \delta u_t + f(x_t^*, u_t^*)$$

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}$$

✓ $x_{t+1} = A_t x_t + B_t u_t$

✓ $x_{t+1} = A x_t + B u_t$

Strategy: Build up on LQR

$$x_{t+1} = \left.\frac{\partial f}{\partial x}\right|_{x_t} \delta x_t + \left.\frac{\partial f}{\partial u}\right|_{u_t} \delta u_t + f(x_t^*, u_t^*)$$

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}$$

✓ $x_{t+1} = A_t x_t + B_t u_t$

✓ $x_{t+1} = A x_t + B u_t$

Strategy: Build up on LQR

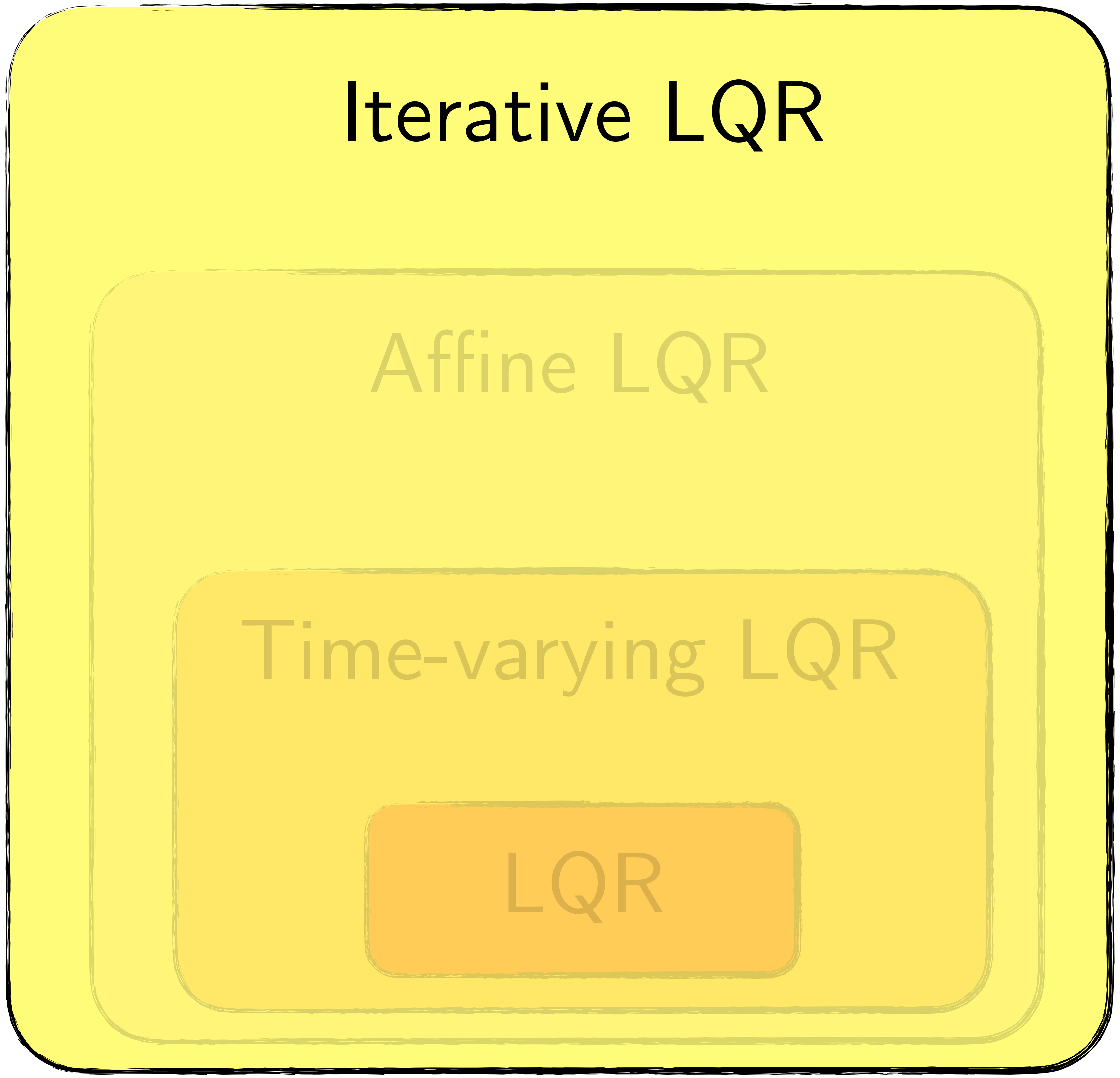
$$x_{t+1} = \frac{\partial f}{\partial x}\bigg|_{x_t} \delta x_t + \frac{\partial f}{\partial u}\bigg|_{u_t} \delta u_t + f(x_t^*, u_t^*)$$

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}$$

$$x_{t+1} = A_t x_t + B_t u_t$$

$$x_{t+1} = A x_t + B u_t$$

Strategy: Build up on LQR



$$x_{t+1} = \left.\frac{\partial f}{\partial x}\right|_{x_t} \delta x_t + \left.\frac{\partial f}{\partial u}\right|_{u_t} \delta u_t + f(x_t^*, u_t^*)$$

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}$$

$$x_{t+1} = A_t x_t + B_t u_t$$

$$x_{t+1} = A x_t + B u_t$$

The iLQR Algorithm

1. Propose some initial (feasible) trajectory $\{x_t, u_t\}_{t=0}^{T-1}$
2. Linearize the dynamics, f about trajectory:

$$\left.\frac{\partial f}{\partial x}\right|_{x_t} = A_t, \quad \left.\frac{\partial f}{\partial u}\right|_{u_t} = B_t$$

   Linearization can be obtained by three methods:

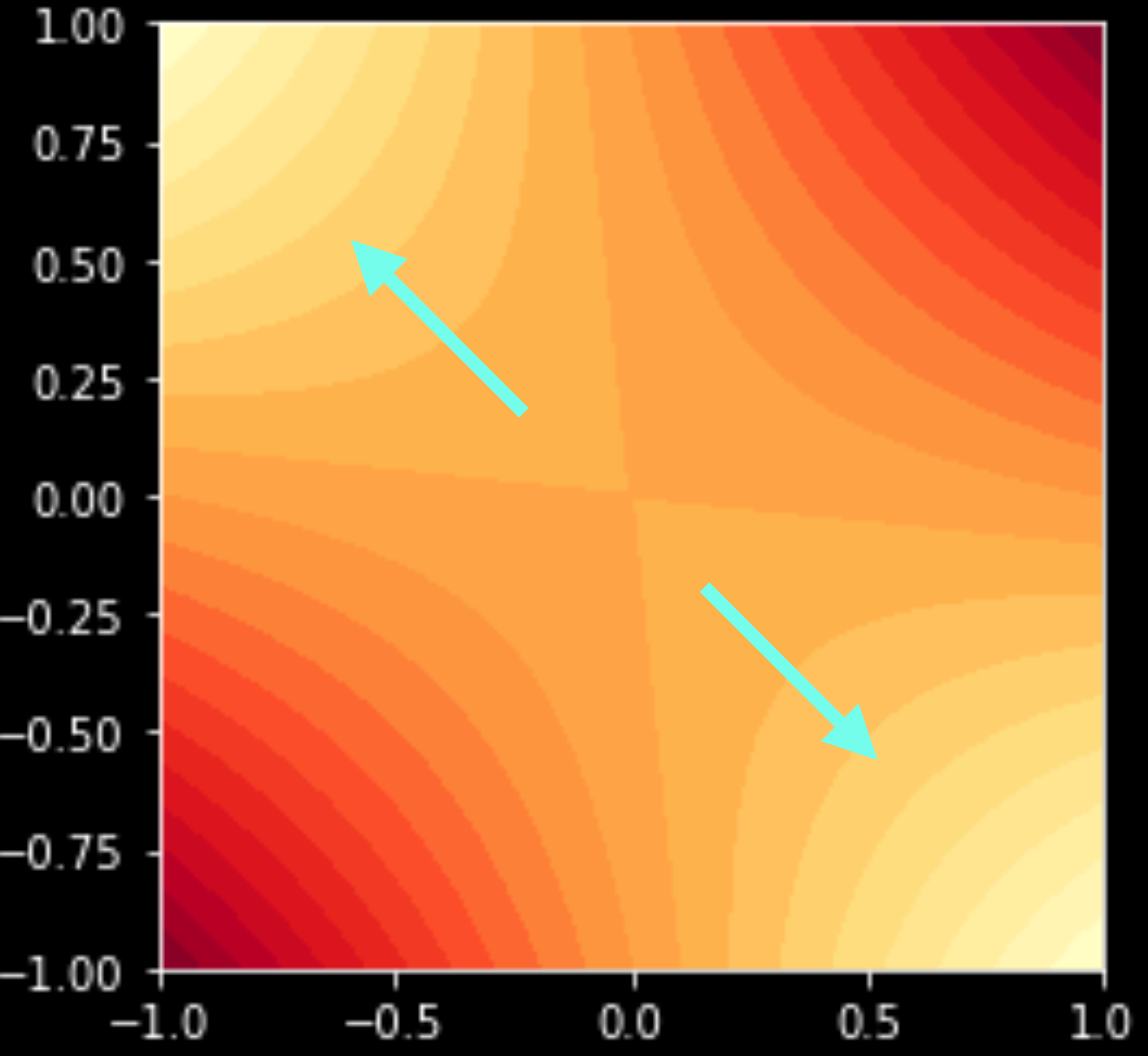
   (a) Analytical: either manually or via *auto-diff*, compute the correct derivatives.
   (b) Numerical: use finite differencing.
   (c) Statistical: Collect samples by deviations around the trajectory and fit linear model.

3. Compute second order Taylor series expansion the cost function $c(x, u)$ around x_t and u_t and get a quadratic approximation $c_t(\tilde{x}_t, \tilde{u}_t) = \tilde{x}_t^\top \tilde{Q}_t \tilde{x}_t + \tilde{u}_t^\top \tilde{R}_t \tilde{u}_t$ where the $\tilde{x}_t, \tilde{u}_t$ variables represent *changes* in the proposed trajectory in homogenous coordinates. [12]
4. Given $\{A_t, B_t, \tilde{Q}_t, \tilde{R}_t\}_{t=0}^{T-1}$, solve an affine quadratic control problem and obtain the proposed feedback matrices (on the homogeneous represenation of x).
5. Forward simulate the full nonlinear model $f(x, u)$ using the computed controls $\{u_t\}_{t=0}^{T-1}$ that arise from feedback matrices applied to the sequence of states $\{x_t\}_{t=0}^{T-1}$ that arise from that forward simulation.
6. Using the newly obtained $\{x_t, u_t\}_{t=0}^{T-1}$ repeat steps from 2.

Approximations always hurt

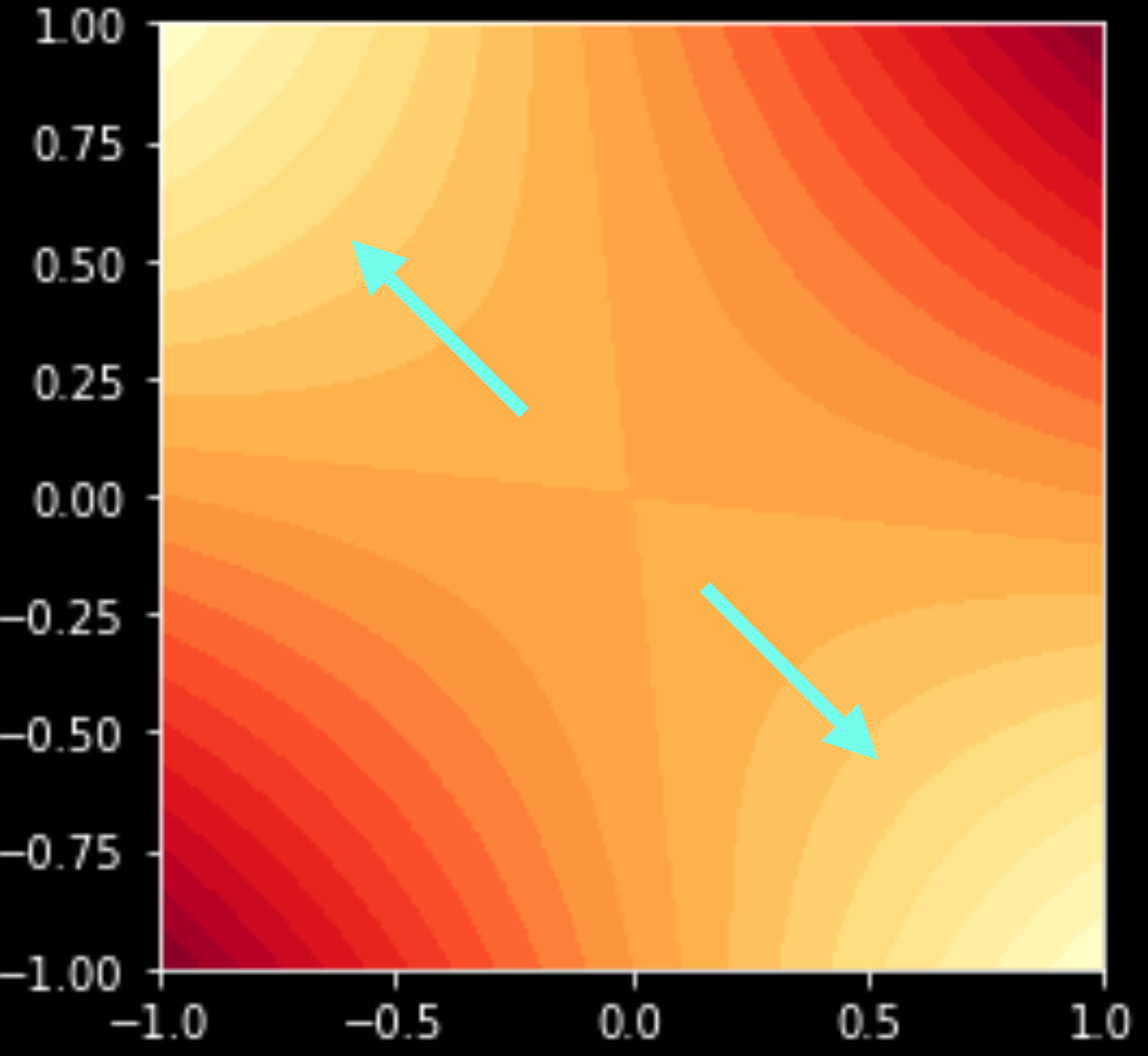
#1: Q and R not PSD / PD

Quadracizing non-convex cost function



#1: Q and R not PSD / PD

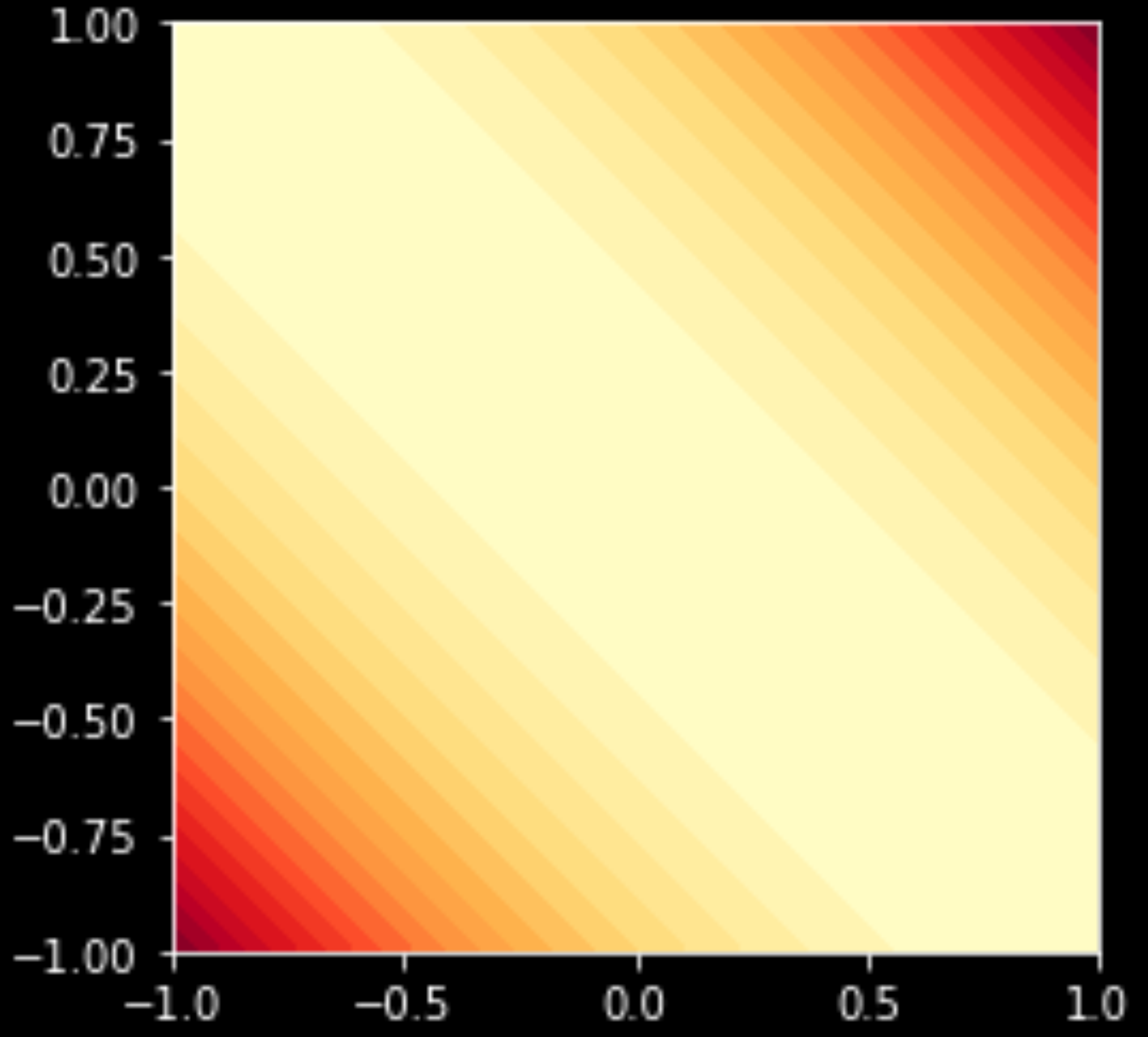
Quadracizing non-convex cost function



Eigen-value decomposition

$$Q = U\Sigma U^T$$

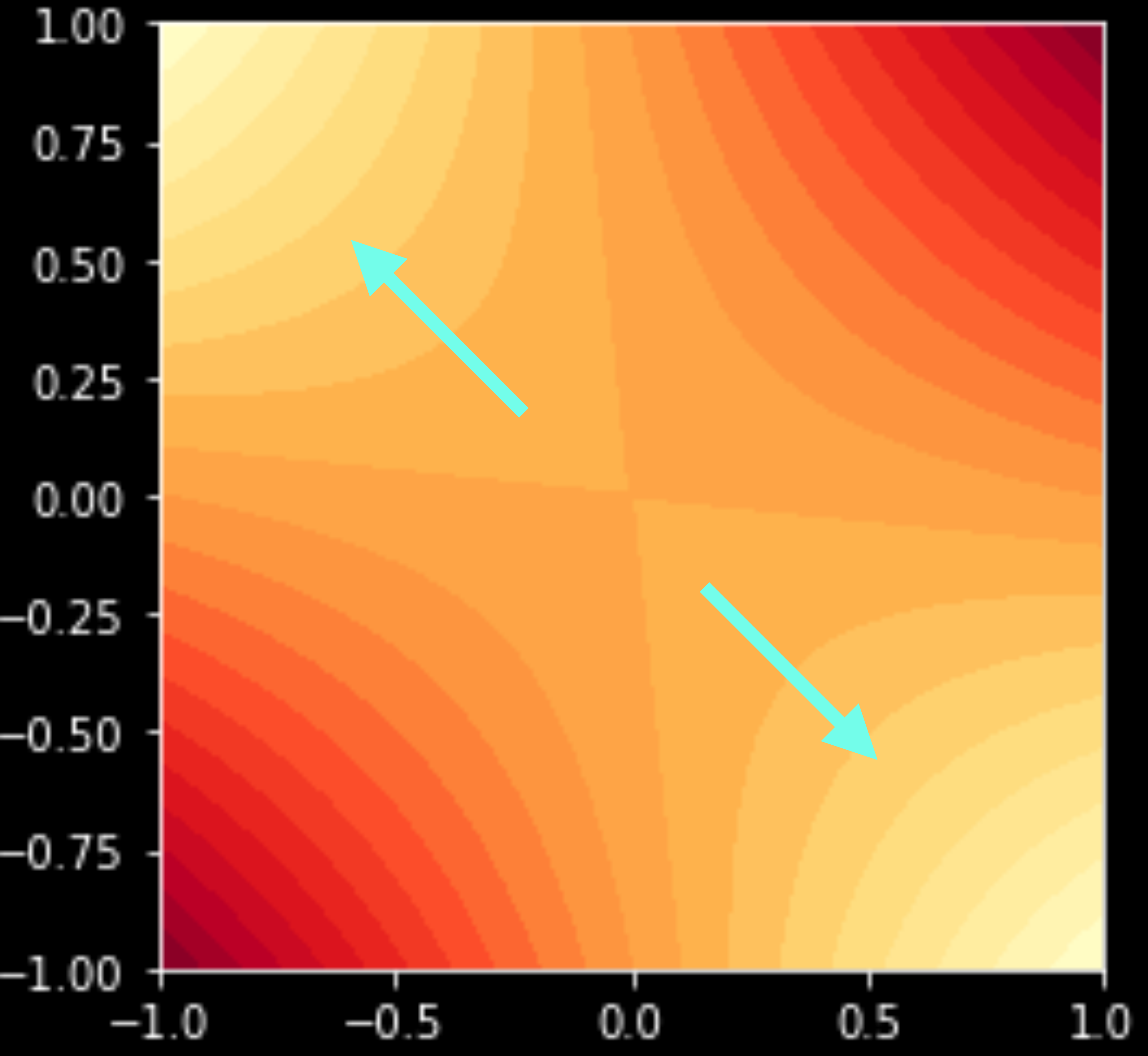
Set negative eigen values to 0



$$\Sigma = \begin{bmatrix} 6 & 0 \\ 0 & -4 \end{bmatrix} \longrightarrow \Sigma = \begin{bmatrix} 6 & 0 \\ 0 & 0 \end{bmatrix}$$

#1: Q and R not PSD / PD

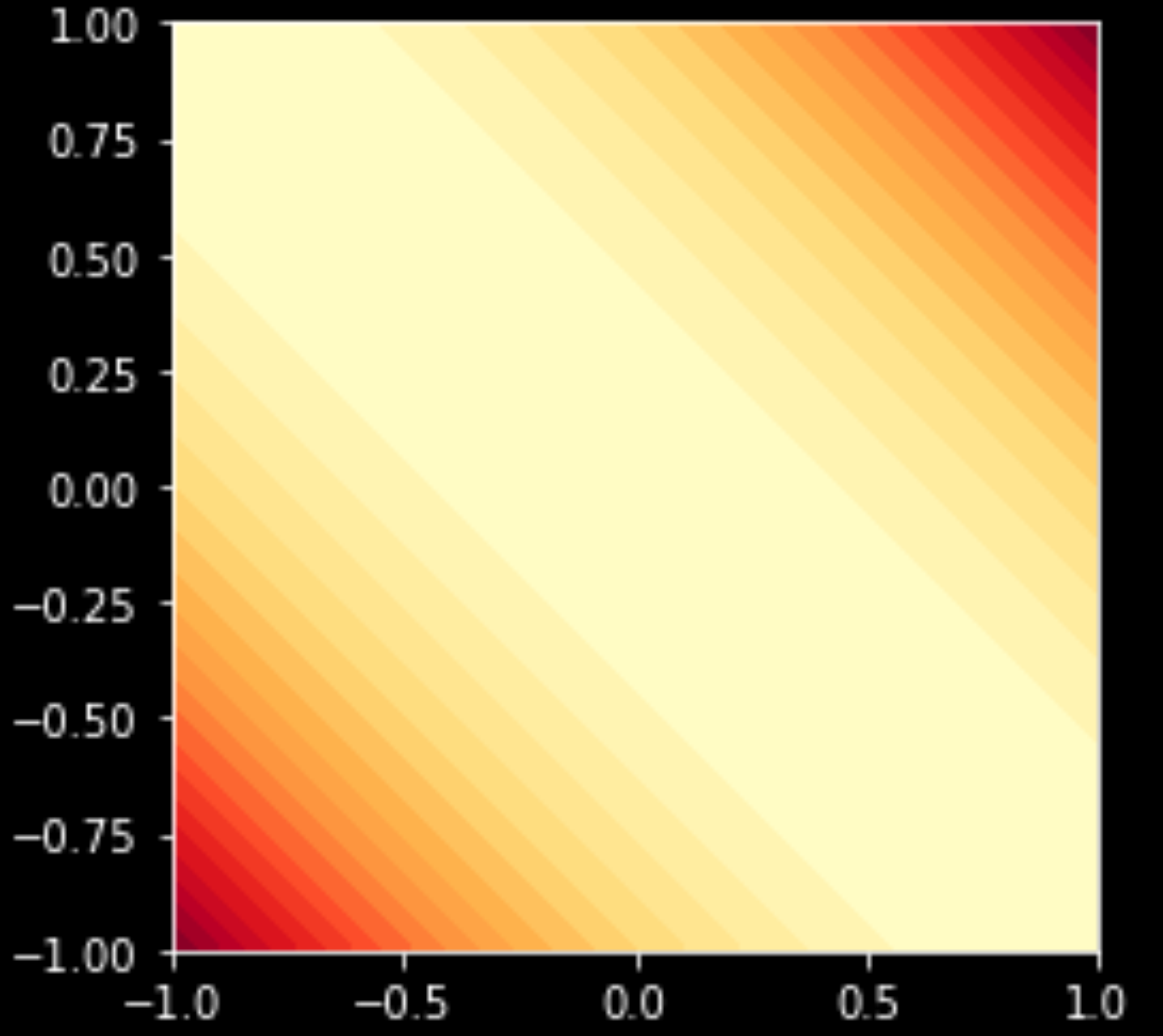
Quadracizing non-convex cost function



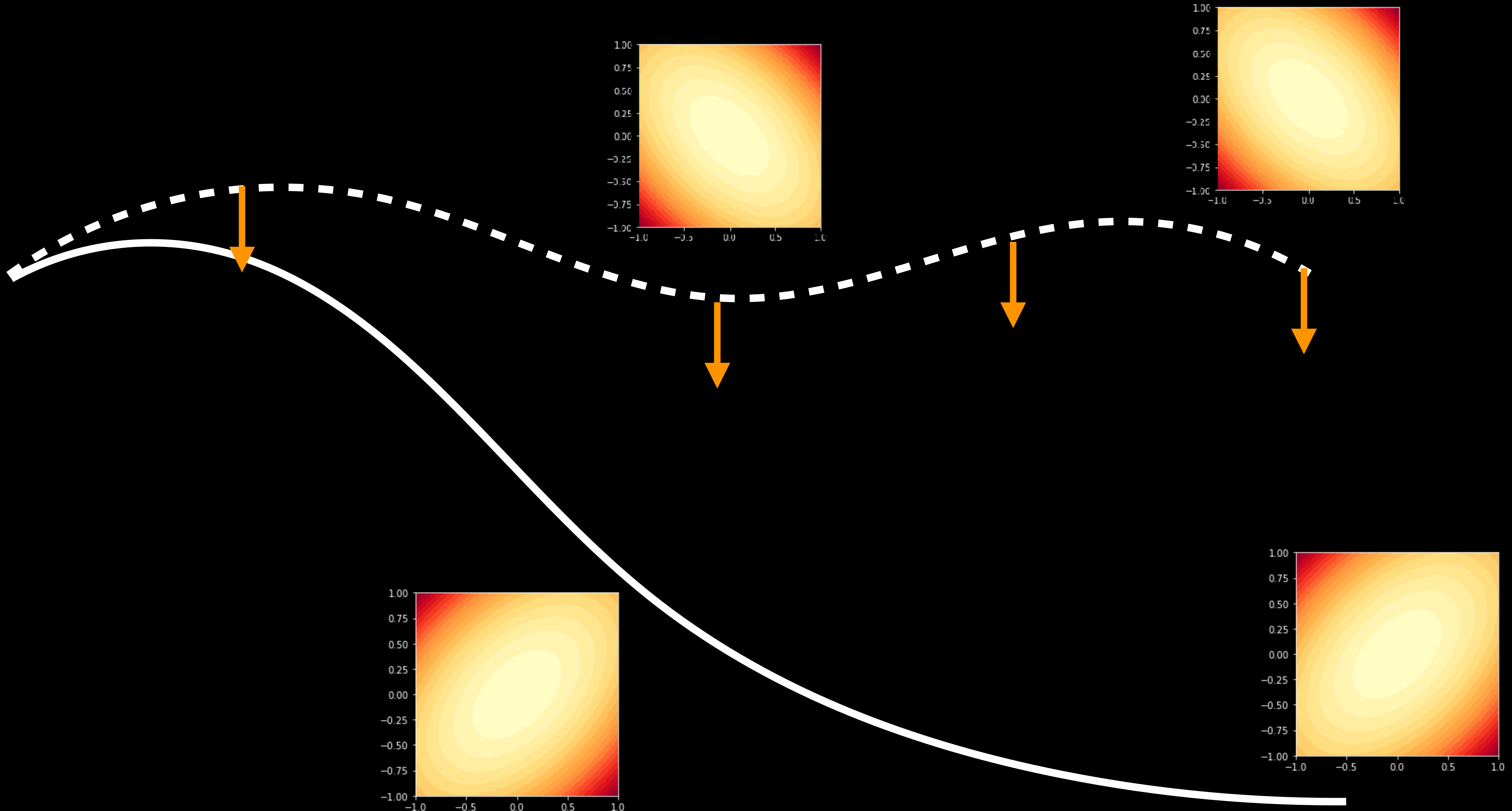
Increase diagonal values

$$Q = Q + \lambda I$$

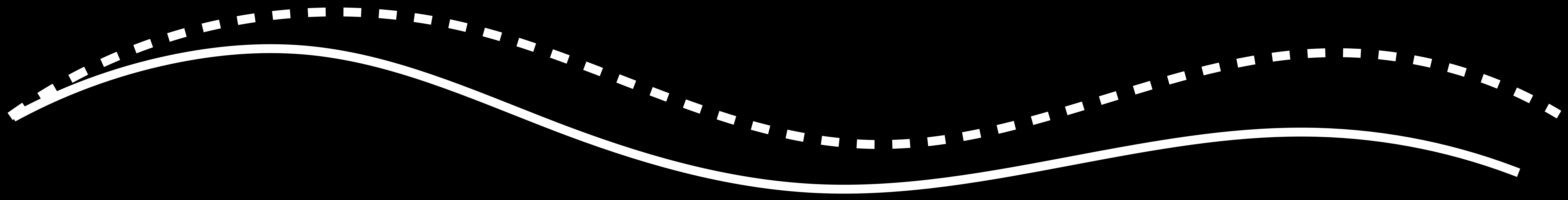
$\lambda = 4$



#2: Approximation Errors Compound



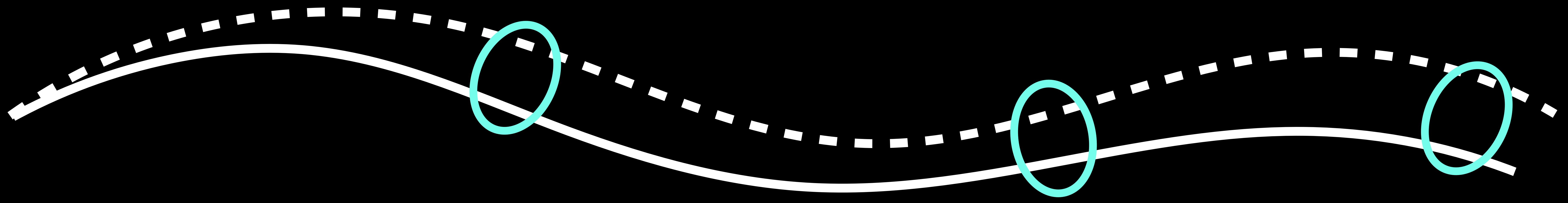
#2: Approximation Errors Compound



Slowly change controls

$$u = (1 - \alpha)u_{old} + \alpha u_{new}$$

#2: Approximation Errors Compound



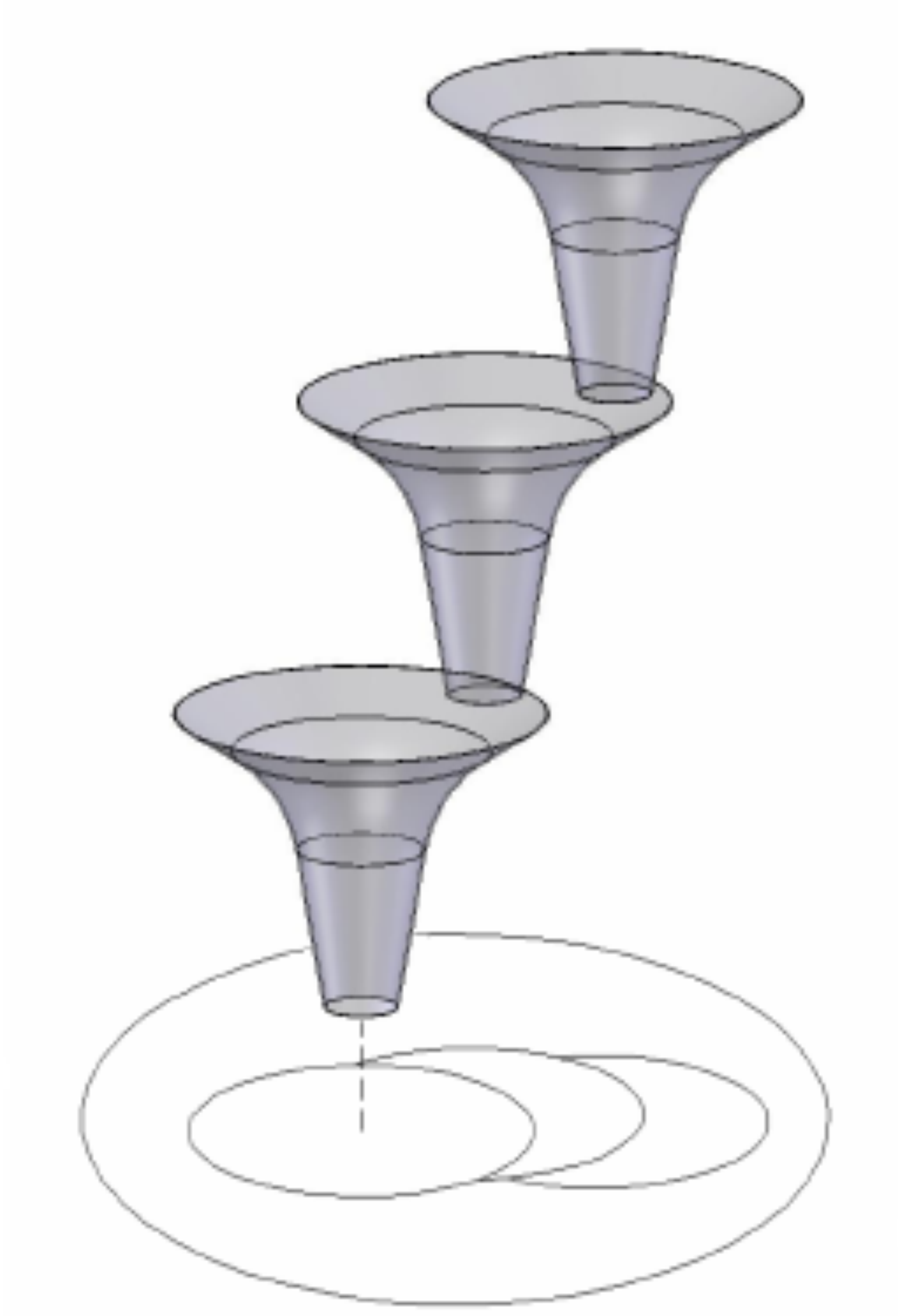
Trust region: Control and state sampling

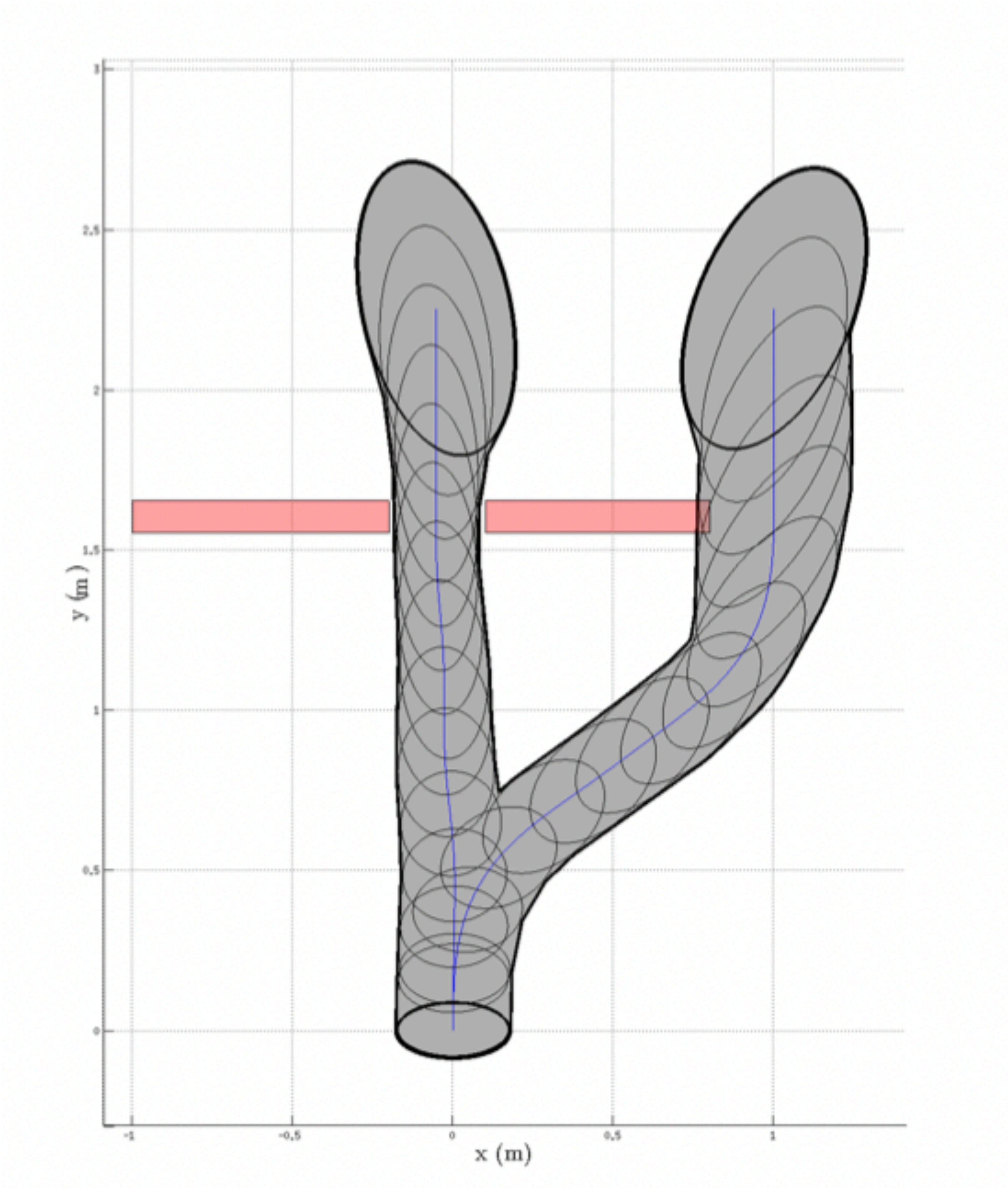
$$c_{new}(x, u) = c(x, u) + \lambda_x ||x - x_{old}|| + \lambda_u ||u - u_{old}||$$

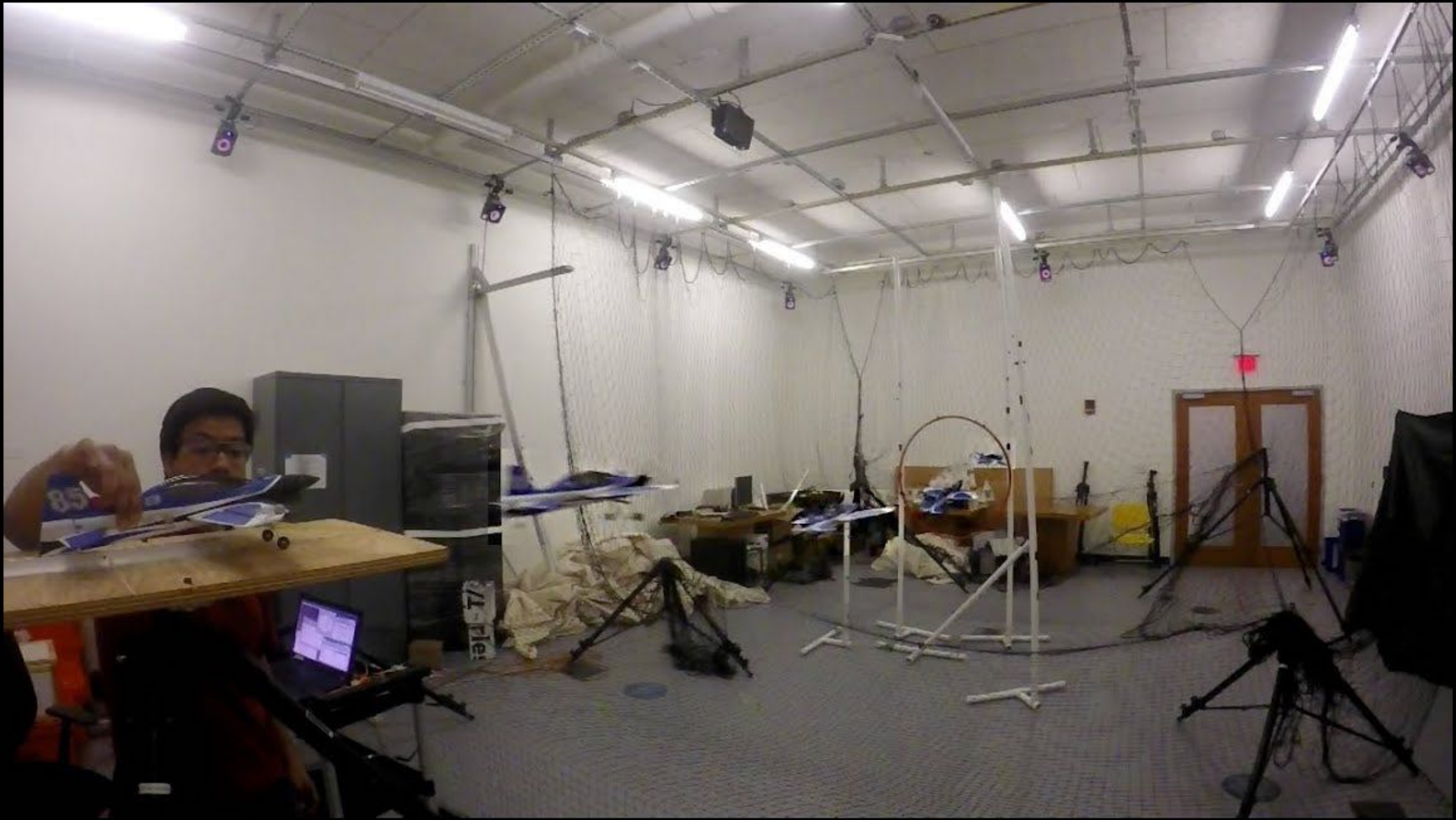
(Penalize deviations from old state / control)

How general
is this idea?

#1: Cover the world with funnels






85

#2: Replace linear/quadratic with a LEARNER

for i = 1 ..... N

Roll-out current policy

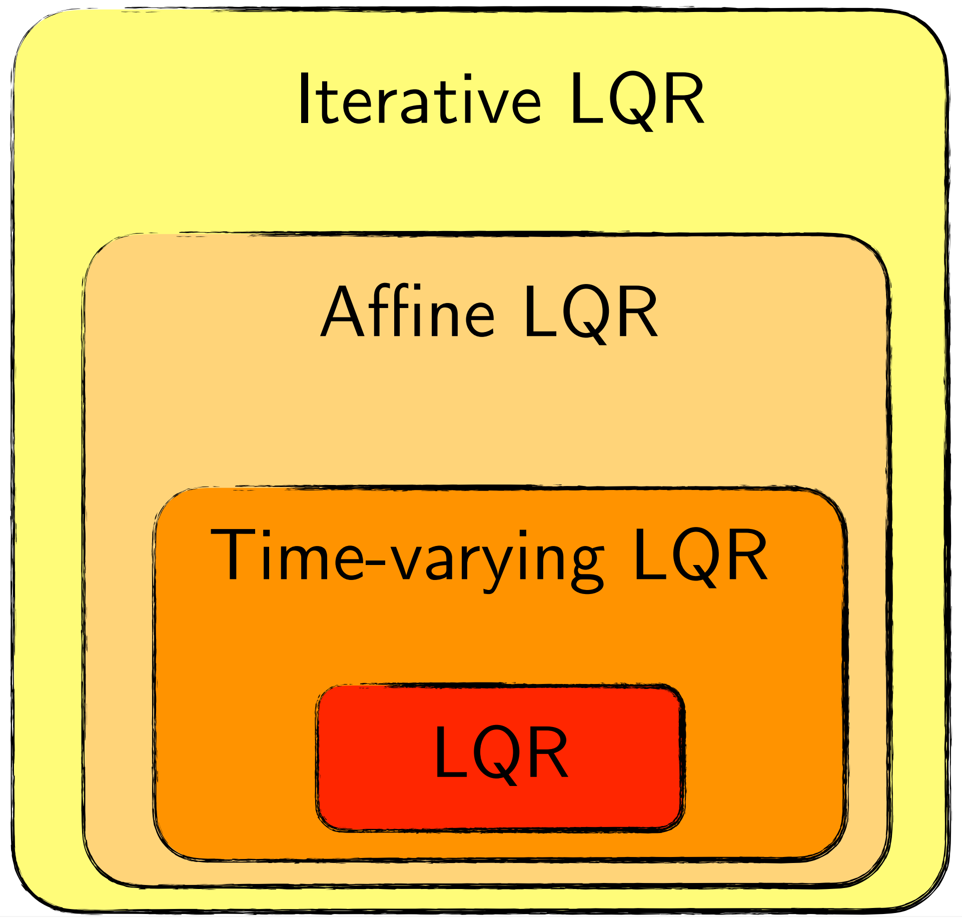
~~Linearize dynamics,~~
~~Quadricize costs about traj~~ → Train model from collected data!

Update policy

tl;dr

LQR is fundamentally a way to *locally approximate* and update value functions

Strategy: Build up on LQR



$$x_{t+1} = \frac{\partial f}{\partial x}\bigg|_{x_t} \delta x_t + \frac{\partial f}{\partial u}\bigg|_{u_t} \delta u_t + f(x_t^*, u_t^*)$$

$$x_{t+1} = A_t x_t + B_t u_t + x_t^{off}$$

$$x_{t+1} = A_t x_t + B_t u_t$$

$$x_{t+1} = A x_t + B u_t$$

Approximations always hurt